Package: geneplotter (via r-universe)

November 1, 2026

Title Graphics related functions for Bioconductor

Version 1.90.0

Description Functions for plotting genomic data

Depends R (>= 2.10), methods, Biobase, BiocGenerics, lattice, annotate

Imports AnnotationDbi, graphics, grDevices, grid, RColorBrewer, stats, utils

Suggests Rgraphviz, fibroEset, hgu95av2.db, hu6800.db, hgu133a.db, BiocStyle, knitr

VignetteBuilder knitr

License Artistic-2.0

biocViews Visualization

LazyLoad yes

Config/pak/sysreqs libpng-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:29:56 UTC

RemoteUrl https://github.com/bioc/geneplotter

RemoteRef RELEASE_3_23

RemoteSha 5aeaf770752f850ef1c8e0af27b21ecf1460397c

Contents

alongChrom 2
amplicon.plot 4
cColor 5
cPlot 6
cScale 7
expressionSet133a 8
GetColor 9
groupedHeatmap 10
histStack 11
imageMap-methods 12

make.chromOrd 14
Makesense 15
multiecdf 16
openHtmlPage 18
plotChr 19
plotExpressionGraph 21
plotMA-methods 22
savepng 23

Index **25**

`alongChrom` *A function for plotting expression data from an ExpressionSet for a given chromosome.*

Description

Given a particular ExpressionSet object, a chromLocation object, and a chromosome name, will plot selected ExpressionSet data using various methods.

Usage

```
alongChrom(eSet, chrom, specChrom,  xlim, whichGenes,
plotFormat=c("cumulative", "local","image"),
xloc=c("equispaced", "physical"),
scale=c("none","zscale","rankscale","rangescale","zrobustscale"),
geneSymbols=FALSE, byStrand=FALSE, colors="red", lty=1, type="S",
...)
```

Arguments

`eSet`	The ExpressionSet object to be used.
`chrom`	The desired chromosome.
`specChrom`	An object of type chromLocation for the species being represented.
`xlim`	A pair of values - either character or integer, which will denote the range of genes to display (based on base pair: either directly in the case of integers, or using the locations of the named genes if character). If not supplied, the entire chromosome is used.
`whichGenes`	If supplied, will limit the displayed genes to the ones provided in this vector.
`xloc`	Determines whether the X axis points (gene names) will be displayed according to their relative position on the chromosome (physical), or spaced evenly (equispaced). Default is equispaced.
`plotFormat`	Determines the method which to plot the data.
`scale`	Determines what method of scaling will be applied to the data. Default is none.
`geneSymbols`	Notes whether to use Affy IDs or Gene Symbols, default is Affy IDs

byStrand	Determines whether to show the entire plot at once, or a split plot by strands. Default is a singular plot
lty	A vector of line types, which will be cycled.
type	Plot type, from par. Defaults to "S".
colors	A vector of colors for the plots, which will be cycled.
...	Any remaining graphics commands may be passed along as per plot()

Details

The genes on the chromosome of interest are extracted from the chromLocation object passed in, which are then intersected with the genes listed in the ExpressionSet. These remaining genes will then be plotted according to the plotFormat argument. If image is specified, an image plot is created showing the expression levels of the samples by gene, using a colour map to denote the levels. If cumulative is chosen, the cumulative expression level is plotted against the genes for each sample. Likewise, if local is used, the raw data is plotted for each sample against the genes using a boxplot format.

Not all parameters are honored for all plotformats. xloc, lty, and type are only used with the cumulative plotformat.

Author(s)

Jeff Gentry

Examples

```
data(sample.ExpressionSet)
## A bit of a hack to not have a package dependency on hgu95av2
## but need to fiddle w/ the warn level to not fail the example anyways.
curWarn <- options(warn=0)
on.exit(options(curWarn), add=TRUE)
if (require("hgu95av2.db")) {
   z <- buildChromLocation("hgu95av2")
   lty <- c(1, 2, 3, 4, 5)
   cols <- c("red", "green", "blue", "orange", "magenta", "black")
   cols <- cols[sample.ExpressionSet$type]
   if (interactive()) {
      par(ask=TRUE)
   }

   ## Here we're using xlim to denote a physical region to display
   xlim <- c(87511280,127717880)
   for (xl in c("equispaced", "physical"))
     for (sc in c("none","rangescale"))
     {
        alongChrom(sample.ExpressionSet, "1", z, xlim=xlim, xloc=xl,
           plotFormat="cumulative", scale=sc,lty=lty, colors=cols)
     }

  ## Here we're looking for specific genes
  which <- c("31540_at","31583_at", "31508_at", "31529_at", "31439_f_at",
```

```
          "31729_at")
 ## Gene "31529_at" does not exist in the current set of genes,
 ## here it demonstrates how genes not available are dropped.
  for (xl in c("equispaced", "physical"))
    for (sc in c("none","rangescale"))
    {
        alongChrom(sample.ExpressionSet, "1", z, which=which, xloc=xl,
           plotFormat="cumulative", scale=sc,lty=lty, col=cols)
     }

  ## Do an image plot
  for (bs in c(TRUE,FALSE))
     alongChrom(sample.ExpressionSet, "1",z, xlim=xlim, plotFormat="image",
               scale="zscale", byStrand=bs)

  ## A boxplot
  for (st in c(TRUE,FALSE))
     alongChrom(sample.ExpressionSet, "1", z, plotFormat="local",
                colors=cols, byStrand=st)
} else print("Example can not be run without the hgu95av2 data package")
```

`amplicon.plot` *Create an amplicon plot*

Description

Given a two-sample test statistic and an ExpressionSet this function plots regions of the genome that are either highly expressed (in red) or have low expression (blue) differentially in the two groups.

Usage

```
amplicon.plot(ESET, FUN, genome)
```

Arguments

ESET	an object of class ExpressionSet
FUN	A two sample test function suitable for esApply.
genome	A character string of the base name for the annotation.

Details

In some genetic studies we are interested in finding regions of the genome where there are a set of highly expressed genes in some subgroup of the population. This set of highly (or lowly) expressed genes is often of great interest. For example in breast cancer the HER–2 gene is on an amplicon. In some patients approximately 5 genes located near HER–2 are all amplified.

These plot should help in the search for such regions.

Value

No value is returned. This function is executed purely for side effect.

Author(s)

Robert Gentleman

See Also

esApply, make.chromOrd

Examples

```
##none yet; takes too long
```

---

cColor *A function for marking specific probes on a cPlot.*

Description

Given a set of probes, will highlight them in the color desired on a plot which has already been created via the function cPlot().

Usage

```
cColor(probes, color, plotChroms, scale=c("relative","max"), glen=0.4,
       ...)
```

Arguments

probes	The probes that are being highlighted.
color	A vector of colors, recycled as necessary, to highlight the probes.
plotChroms	An object of type chromLocation which contains all the gene information to be plotted.
scale	Whether to plot the graph scaled absolutely or relative by chromosome. Default is absolute.
glen	The length of the gene line plotted.
...	Additional graphics arguments, passed to segments, which is used to draw the vertical ticks.

Details

It is important to call the function cPlot() first. This function will then search for the specific locations of the probes desired, which are contained within the plotChroms instance of a chromLocation class. It will then pass these on to the plotting routine to highlight the desired locations. NOTE: It is important that plotChroms, scale and glen parameters are the same as used for cPlot().

Author(s)

Jeff Gentry

See Also

cPlot, chromLocation-class

Examples

```
if (require("hgu95av2.db")) {
  z <- buildChromLocation("hgu95av2")
  cPlot(z)
  probes <- c("266_s_at", "31411_at", "610_at", "failExample")
  cColor(probes, "red", z)
  probes2 <- c("960_g_at", "41807_at", "931_at", "39032_at")
  cColor(probes2, "blue", z)
} else
  print("Need hgu95av2.db data package for the example")
```

---

cPlot *A plotting function for chromosomes.*

Description

Given a chromLocation object, will plot all the gene locations from that object.

Usage

```
cPlot(plotChroms, useChroms=chromNames(plotChroms),
      scale=c("relative","max"), fg="white", bg="lightgrey",
      glen=0.4, xlab="", ylab="Chromosome",
      main = organism(plotChroms), ...)
```

Arguments

plotChroms	An object of type chromLocation which contains all the gene information to be plotted.
useChroms	A vector of chromosome names to be used in the plot. Default is to use all the chromosomes from the plotChroms object.
scale	Passed on to cScale as it's scale argument. Determines whether the graph is scaled on a relative or absolute basis.
fg	The colour to be used for the genes. Default is white.
bg	The colour to be used for the background of the plot. Defaults to lightgrey.
glen	A scaling factor applied to the plotted length of each gene. Defaults to 0.4 - it is recommended that this not be set larger then 0.5 as it will cause overlap between chromosomes.

xlab	A label for the x axis.
ylab	A label for the y axis.
main	A main label for the plot.
...	Additional graphics arguments, passed to segments, which is used to draw the vertical ticks.

Details

This function will first use the lengths of the chromosomes, stored in the object to create scaling factors for the X axis. Once the scaling factors are determined, the chromLocation object which is passed in is used to determine all the gene locations/strand information/etc, which is then plotted for the user.

Author(s)

Jeff Gentry

See Also

cScale, cColor, chromLocation-class

Examples

```
## A bit of a hack to not have a package dependency on hgu95av2
## but need to fiddle w/ the warn level to not fail the example anyways.

curWarn <- options(warn=0)
on.exit(options(curWarn), add=TRUE)
if (require("hgu95av2.db")) {
    z <- buildChromLocation("hgu95av2")

    if (interactive()) {
        curPar <- par(ask=TRUE)
        on.exit(par(curPar), add=TRUE)
    }

    for (sc in c("max","relative"))
        cPlot(z,c("1","5","10","X","Y"),sc)
} else print("This example can not be run without hgu95av2 data package")
```

---

cScale *A function for mapping chromosome length to a number of points.*

Description

Given a number of points (generally representing the number of points on a plot's axis), and a vector of chromosome lengths - will generate a vector of the same length as the one passed in containing scaling factors for each chromosome.

Usage

```
cScale(points, cLengths, method=c("max", "relative"), chrom)
```

Arguments

points	The number of points to scale the chromosome length to.
cLengths	A vector of chromosome lengths.
method	Determines whether to use relative or absolute scaling. Default is "max" (absolute).
chrom	Which chrom to determine the scale for

Details

The scale factor is calculated in a manner based on the method argument. If method is max, the factor is derived by dividing the points argument by each chromosome's length (in base pairs). If the method chosen is relative, then the scale is determined by dividing the points argument by the maximum chromsome length, and applying that value to each chromosome.

Author(s)

Jeff Gentry

See Also

cPlot

Examples

```
## A bit of a hack to not have a package dependency on hgu95av2
## but need to fiddle w/ the warn level to not fail the example anyways.
curWarn <- options(warn=0)
on.exit(options(warn), add=TRUE)
if (require("hgu95av2.db")) {
   z <- buildChromLocation("hgu95av2")

  for (sc in c("max","relative"))
    scale <- cScale(1000, chromLengths(z),sc,"Y")
} else print("This example needs the hgu95av2 data package")
```

---

expressionSet133a *A small dataset for testing*

Description

An artificial Affymetrix hgu133a dataset, with one covariate 'cov1'.

Usage

```
data(expressionSet133a)
```

Format

The data are artifical. There are 6 cases labeled 1 to 6 and and 22283 genes as in an Affymetrix U133a chips. There is one covariate (factor) whose values are "type 1" for the first 3 samples and "type 2" for the last 3 samples.

Examples

```
data(expressionSet133a)
```

GetColor *A function to get the Red-Blue color scheme used by dChip*

Description

A simple, vectorized function that computes a Red/Blue color for plotting microarray expression data.

Usage

```
GetColor(value, GreenRed=FALSE, DisplayRange=3)
dChip.colors(n)
greenred.colors(n)
```

Arguments

value	The vector of expression values.
GreenRed	If TRUE the Green-Red colors are produced, otherwise Red-Blue are procduced.
DisplayRange	A parameter controlling the range of value's that will be plotted.
n	An integer saying how many colors to be in the palette.

Details

GetColor is a simple mapping into RGB land provided by Cheng Li. dChip.colors provides functionality similar to that of topo.colors for the red–blue colors used for genome plots. greenred.colors does the same for the green-black-red gradient.

Value

A vector of RGB colors suitable for plotting in R.

Author(s)

R. Gentleman, based on an original by C. Li.

Examples

```
set.seed(10)
x <- rnorm(10)
GetColor(x)
dChip.colors(10)
```

---

groupedHeatmap *Heatmap of a matrix with grouped rows and columns*

Description

The function uses grid.rect and grid.rect to draw a heatmap with grouped rows and columns.

Usage

```
groupedHeatmap(z, frow, fcol,
 fillcolours = c("#2166ac","#4393c3","#92c5de","#d1e5f0","#fefefe","#fddbc7","#f4a582","#d6604d","#b
  bordercolour = "#e0e0e0",
  zlim = range(z, na.rm=TRUE))
```

Arguments

z	A matrix with row and column names.
frow	A factor of length nrow(z) indicating the row grouping.
fcol	A factor of length ncol(z) indicating the column grouping.
fillcolours	A character vector of colours from which the colour map is obtained through interpolation.
bordercolour	Either a character vector of length 1, specifying the border colour of the heatmap tiles, or NULL or NA, which indicates that the border colour should match the fill colour.
zlim	Lower and upper limit of z values represented in the colour map.

Details

The function can be called within other drawing operations from the grid package, e.g. within a viewport.

Value

The function is called for its side effect, drawing text and rectangles on the current viewport.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

See Also

grid.text, grid.rect

Examples

```
data("mtcars")

groupedHeatmap(
  scale(mtcars),
  frow = factor(sapply(strsplit(rownames(mtcars), " "), "[", 1)),
  fcol = factor(round(seq_len(ncol(mtcars))/3)))
```

---

histStack *Stacked histogram*

Description

Stacked histogram

Usage

```
histStack(x, breaks, breaksFun=paste, ylab="frequency", ...)
```

Arguments

x	A list of numeric vectors.
breaks	Histogram breaks, as in hist
breaksFun	Function, can be used to control the formatting of the bin labels. See example.
ylab	Label for the Y-axis on the plot
...	Further arguments that get passed to barplot

Details

The function calls hist for each element of x and plots the frequencies as a stacked barplot using barplot with beside=FALSE.

Value

The function is called for its side effect, producing a barplot on the active graphics device. It returns the result of the call to barplot.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

Examples

```
x    <- list(rnorm(42), rnorm(42)+2)
br   <- seq(-3, 5, length=13)
cols <- c("#1D267B", "#ceffc0")
histStack(x, breaks=br, col=cols)

histStack(x, breaks=br, col=cols,
          breaksFun=function(z) paste(signif(z, 3)))
```

`imageMap-methods` *Write an HTML IMG tag together with a MAP image map.*

Description

Write an HTML IMG tag together with a MAP image map.

Usage

```
  ## S4 method for signature 'matrix,connection,list,character'
imageMap(object, con, tags, imgname)
```

Arguments

`object`	Matrix with 4 columns, specifying the coordinates of the mouse-sensitive region . Each row specifies the corners of a rectangle within the image, in the following order: (left x, lower y, right x, upper y). Note that the point (x=0, y=0) is at the left upper side of the image.
`con`	Connection to which the image map is written.
`tags`	Named list whose elements are named character vectors. Names must correspond to node names in `object`. See details.
`imgname`	Character. Name of the image file (for example PNG file) that contains the plot.

Details

The most important tags are `TITLE`, `HREF`, and `TARGET`. If the list `tags` contains an element with name `TITLE`, then this must be a named character vector containing the tooltips that are to be displayed when the mouse moves over a node. The names of the nodes are specified in the `names` attribute of the character vector and must match those of `object`.

Similarly, `HREF` may be used to specify hyperlinks that the browser can follow when the mouse clicks on a node, and `TARGET` to specify the target browser window.

Currently, only rectangular regions are implemented; the actual shape of the nodes as specified in `object` is ignored. Also, tags for edges of the graph are currently not supported.

This function is typically used with the following sequence of steps:

1. generate your graphic and save it as a bitmap file, e.g. using the `jpeg`, `png`, or `bitmap` device. At this stage, you also need to figure out the pixel coordinates of the interesting regions within your graphic. Since the mapping between device coordinates and pixel coordinates is not obvious, this may be a little tricky. See the examples below, and for a more complex example, see the source code of the function `plotPlate`.
2. open an HTML page for writing and write HTML header, e.g. using the `openHtmlPage` function.
3. Call the `imageMap` function.
4. Optionally, write further text into the HTML connection.
5. Close HTML file, e.g. using the `closeHtmlPage` function.

Value

The function is called for its side effect, which is writing text into the connection `con`.

Author(s)

Wolfgang Huber http://www.dkfz.de/abt0840/whuber

See Also

`plotPlate`, `writeLines`

Examples

```
f1  = paste(tempfile(), ".html", sep="")
f2  = paste(tempfile(), ".html", sep="")
fpng = tempfile()

if(capabilities()["png"]) {
  ## create the image
 colors = c("#E41A1C","#377EB8","#4DAF4A","#984EA3","#FF7F00","#FFFF33","#A65628","#F781BF","#999999")
  width  = 512
  height = 256
  png(fpng, width=width, height=height)
  par(mai=rep(0,4))
  plot(0,xlim=c(0,width-1),ylim=c(0,height-1),xaxs="i",yaxs="i",type="n",bty="n")
  cx=floor(runif(100)*(width-11))
  cy=floor(runif(100)*(height-11))
  coord=cbind(cx, cy, cx+10, cy+10)
  rect(coord[,1], height-coord[,2], coord[,3], height-coord[,4],
       col=sample(colors, 100, replace=TRUE))
  text(width/2, height-3, "Klick me!", adj=c(0.5, 1), font=2)
  dev.off()

  ## create the frame set
  cat("<html><head><title>Hello world</title></head>\n",
      "<frameset rows=\"280,*\" border=\"0\">\n",
      "<frame name=\"banner\" src=\"file://", f2, "\">\n",
      "<frame name=\"main\" scrolling=\"auto\">",
```

```
    "</frameset>", sep="",file=f1)

  ## create the image map
  href  =sample(c("www.bioconductor.org", "www.r-project.org"),nrow(coord),replace=TRUE)
 title =sample(as.character(packageDescription("geneplotter")),nrow(coord),replace=TRUE)
  con = file(f2, open="w")
  imageMap(coord, con,
    list(HREF=paste("http://", href, sep=""),
         TITLE=title, TARGET=rep("main", nrow(coord))), fpng)
  close(con)

  cat("Now have a look at file ", f1, " with your browser.\n", sep="")
}
```

---

make.chromOrd *Make a chromOrd object*

Description

This function makes a chromOrd object.

Usage

```
make.chromOrd(genome, gnames)
```

Arguments

genome	A character string.
gnames	A character vector of the genes to be selected.

Details

This function reads in a lot of annotation data and creates a list with one element for each chromosome. The elements of this list are indices indicating the order of the genes that are on that chromosome (and in the annotation data set being used).

Value

A list of chromOrd type. One element for each chromosome. Suitable for reordering other values according to the chromosomal location.

Author(s)

Robert Gentleman

See Also

amplicon.plot

Examples

```
data(sample.ExpressionSet)
make.chromOrd("hgu95A", featureNames(sample.ExpressionSet))
```

Makesense *Produce Smoothed Sense/Anti-sense For All Chromosomes*

Description

'Makesense' takes either an ExpressionSet object or a matrix of gene expressions and will produce a smoothed positive and negative strands for all chromosomes.

Usage

```
Makesense(expr, lib, ...)
```

Arguments

expr	Either an ExpressionSet or a matrix of gene expressions with genes as rows and columns as samples.
lib	The name of the Bioconductor annotation data package that will be used to provide mappings from probes to chromosomal locations, such as hgu95av2.db or hgu133a.db. If expr is an ExpressionSet, the argument defaults to the annotation slot of the ExpressionSet.
...	Currently, the only optional argument is f, the smoother span to be passed to 'lowess'. Its value should be in the interval of (0,1). This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness. The default value for this argument is 1/10.

Details

The expr argument can either be of class ExpressionSet or matrix, where the latter represents the matrix of gene expressions.

If the expr argument is an ExpressionSet, the lib argument will use the annotation slot. Users can override this behaviour and supply their own lib argument if they wish. If the ExpressionSet has no value associated with the annotation slot (which should not happen, but is possible) then the user must supply the lib argument manually or the function will throw an error.

Value

A list of 2 components:

ans2	a list, whose components correspond to samples in the same order as appearing in the columns of 'expr'. Each component is also a list, named by chromosomes "1"-"22", "X" and "Y". Each named component is again a list with two elements named "posS" and "negS", corresponding to the positive and negative strands of a chromosome, each of which is an object returned by 'lowess'.
lib	A string giving the name of the annotation data package to use. Optional if expr is an ExpressionSet.

Author(s)

Robert Gentleman and Xiaochun Li

See Also

plotChr

Examples

```
if (require("hgu133a.db")) {
  data(expressionSet133a)
  esetobj <- Makesense(exprs(expressionSet133a), "hgu133a")
  esetobj2 <- Makesense(expressionSet133a[1:200, ])
}
```

`multiecdf` *Multiple empirical cumulative distribution functions (ecdf) and densities*

Description

Plot multiple empirical cumulative distribution functions (ecdf) and densities with a user interface similar to that of boxplot. The usefulness of `multidensity` is variable, depending on the data and the smoothing kernel. `multiecdf` will in many cases be preferable. Please see Details.

Usage

```
multiecdf(x, ...)
## S3 method for class 'formula'
multiecdf(formula, data = NULL, xlab, na.action = NULL, ...)
## S3 method for class 'matrix'
multiecdf(x, xlab, ...)
## S3 method for class 'list'
multiecdf(x,
          xlim,
          col = brewer.pal(9, "Set1"),
          main = "ecdf",
          xlab,
          do.points = FALSE,
          subsample = 1000L,
          legend = list(
            x = "right",
            legend = if(is.null(names(x))) paste(seq(along=x)) else names(x),
            fill = col),
          ...)

multidensity(x, ...)
```

```
## S3 method for class 'formula'
multidensity(formula, data = NULL, xlab, na.action = NULL, ...)
## S3 method for class 'matrix'
multidensity(x, xlab, ...)
## S3 method for class 'list'
multidensity(x,
             bw = "nrd0",
             xlim,
             ylim,
             col  = brewer.pal(9, "Set1"),
             main = if(length(x)==1) "density" else "densities",
             xlab,
             lty  = 1L,
             legend = list(
               x = "topright",
              legend = if(is.null(names(x))) paste(seq(along=x)) else names(x),
               fill = col),
             density = NULL,
             ...)
```

Arguments

formula	a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).
data	a data.frame (or list) from which the variables in formula should be taken.
na.action	a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
x	methods exist for: formula, matrix, data.frame, list of numeric vectors.
bw	the smoothing bandwidth, see the manual page for density. The length of bw needs to be either 1 (in which case the same is used for all groups) or the same as the number of groups in x (in which case the corresponding value of bw is used for each group).
xlim	Range of the x axis. If missing, the data range is used.
ylim	Range of the y axis. If missing, the range of the density estimates is used.
col, lty	Line colors and line type.
main	Plot title.
xlab	x-axis label.
do.points	logical; if TRUE, also draw points at the knot locations.
subsample	numeric or logical of length 1. If numeric, and larger than 0, subsamples of that size are used to compute and plot the ecdf for those elements of x with more than that number of observations. If logical and TRUE, a value of 1000 is used for the subsample size.
legend	a list of arguments that is passed to the function legend.
density	a list of arguments that is passed to the function density.
...	Further arguments that get passed to the plot functions.

Details

Density estimates: `multidensity` uses the function `density`. If the density of the data-generating process is smooth on the real axis, then the output from this function tends to produce results that are good approximations of the true density. If, however, the true density has steps (this is in particular the case for quantities such as p-values and correlation coefficients, or for some distributions that have weight only on the posititve numbers, or only on integer numbers), then the output of this function tends to be misleading. In that case, please either use `multiecdf` or histograms, or try to improve the density estimate by setting the `density` argument (`from`, `to`, `kernel`).

Bandwidths: the choice of the smoothing bandwidths in `multidensity` can be problematic, in particular, if the different groups vary with respect to range and/or number of data points. If curves look excessively wiggly or overly smooth, try varying the arguments `xlim` and `bw`; note that the argument `bw` can be a vector, in which case it is expect to align with the groups.

Value

For the `multidensity` functions, a list of `density` objects.

Author(s)

Wolfgang Huber

See Also

`boxplot`, `ecdf`, `density`

Examples

```
words = strsplit(packageDescription("geneplotter")$Description, " ")[[1]]
factr = factor(sample(words, 2000, replace = TRUE))
x = rnorm(length(factr), mean=as.integer(factr))

multiecdf(x ~ factr)
multidensity(x ~ factr)
```

---

openHtmlPage *Open and close an HTML file for writing.*

Description

Open and close an HTML file for writing..

Usage

```
openHtmlPage(name, title="")
closeHtmlPage(con)
```

Arguments

name	Character. File name (*without* the extension '.html').
title	Character. Value of the title tag in the HTML header.
con	Connection.

Details

See example.

Value

For openHtmlPage, a connections.

Author(s)

Wolfgang Huber http://www.dkfz.de/abt0840/whuber

Examples

```
fn <- tempfile()
con <- openHtmlPage(fn, "My page")
writeLines("Hello world", con)
closeHtmlPage(con)
readLines(paste(fn, ".html", sep=""))
```

---

plotChr *Plot Smoothed Sense/Anti-sense of Specified Chromosomes*

Description

For a given chromosome, plot the smooths of the sense and the anti-sense from 5' to 3' (left to right on x-axis).

Usage

```
plotChr(chrN, senseObj, cols = rep("black", length(senseObj[[1]])), log = FALSE, xloc = c("equispaced",
```

Arguments

chrN	The desired chromosome, e.g. for humans it would be a character string in the set of c(1:22, "X", "Y").
senseObj	The result of Makesense.
cols	A vector of colors for the lines in the plot, typically specified according to a certain pheotype of samples.
log	Logical, whether log-transformation should be taken on the smoothed expressions.

xloc	Determines whether the "Representative Genes" will be displayed according to their relative positions on the chromosome (physical), or spaced evenly (equispaced). Default is equispaced.
geneSymbols	Logical, whether to use Affy IDs or Gene Symbols for "Representative Genes", default is Affy IDs.
ngenes	Desired number of "Representative Genes". The number of actual displayed genes may differ.
lines.at	A vector of Affy IDs. Vertical lines will be drawn at specified genes.
lines.col	A vector of colors associated with lines.at.

Author(s)

Robert Gentleman and Xiaochun Li

See Also

Makesense

Examples

```
example(Makesense)

if (interactive())
  op <- par(ask=TRUE)

cols <- ifelse(expressionSet133a$cov1=="test 1", "red", "green")
plotChr("21", esetobj, cols)

# plot on log-scale:

plotChr("21", esetobj, cols, log=TRUE)

# genesymbol instead of probe names:

plotChr("21", esetobj, cols, log=TRUE, geneSymbols=TRUE)

# add vertical lines at genes of interest:

gs <- c("220372_at", "35776_at", "200943_at")
plotChr("21", esetobj, cols, log=TRUE, geneSymbols=FALSE, lines.at=gs)

# add vertical lines at genes of interest
# with specified colors:

gs <- c("220372_at", "35776_at", "200943_at")
cc <- c("blue", "cyan","magenta")
plotChr("21", esetobj, cols, log=TRUE, geneSymbols=FALSE, lines.at=gs,
lines.col=cc)
if (interactive())
  par(op)
```

plotExpressionGraph *A function to plot a graph colored by expression data*

Description

Given a graph and expression data for one entity, will plot the graph with the nodes colored according to the expression levels provided.

Usage

```
plotExpressionGraph(graph, nodeEGmap, exprs, ENTREZIDenvir, mapFun, log = FALSE, nodeAttrs = list(), ..
```

Arguments

graph	The graph to plot
nodeEGmap	A list with element names being node names and the elements being EntrezLink IDs corresponding to those node names.
exprs	A vector of expression data, with names being Affymetrix IDs and values being the expression level.
ENTREZIDenvir	An environment mapping Affymetrix IDs to EntrezLink IDs, such as the ones provided in the xxx2ENTREZID environments from the Bioconductor data packages (where xxx) is a data package).
mapFun	A function to map expression levels to colors.
log	Whether or not the expression data.
nodeAttrs	A list of node attributes, as per plot.graph.
...	Any extra arguments to be passed to plot.graph.

Details

This function can be used to plot a graph and have the nodes colored according to expression levels provided by the user. The graph parameter is a graph object from the graph package.

The nodeEGmap parameter is a list that maps the nodes of the graphs to EntrezLink IDs. An example of this is the IMCAEntrezLink object in the integrinMediatedCellAdhesion data set in the graph package.

The exprs argument is a vector mapping expression levels to Affymetrix IDs. One way to generate an appropriate vector is to extract a single column from an ExpressionSet.

The ENTREZIDenvir environment maps Affymetrix IDs to EntrezLink IDs. The simplest way to provide this argument is to load the preferred Bioconductor data package (e.g. hgu95av2.db) and pass in that package's xxx2ENTREZID, where xxx is the name of the package.

The mapFun function defaults to the function defMapFun, which maps nodes to be either blue, green or red depending for expression ranges of 0-100, 101-500, and 501+. In the case where log is TRUE these ranges are modified with log2. Custom versions of this function can be supplied by the user - it must take two parameters, first the expression vector and a boolean value (log) specifying if the

data has had a log2 applied to it. The function must return a vector with the same names as the expression vector, but the values of the vector will be color strings.

The nodeAttrs list can be specified if any other node attributes are desired to be set by the user. Please see the plot.graph man page for more information on this. The other attribute list (attrs and edgeAttrs) can be passed in via the ... parameter.

The IMCAEntrezLink data structure was created for the purpose of illustrating this program. On Sept 24 2007, the current version of hgu95av2.db was used to map from the nodes of IMCAGraph (in graph package) to Entrez identifiers.

Author(s)

Jeff Gentry

See Also

plot.graph, integrinMediatedCellAdhesion

Examples

```
if (require("Rgraphviz") && require("hgu95av2.db") &&
    require("fibroEset")) {
    data(integrinMediatedCellAdhesion)
 data(IMCAEntrezLink)
    data(fibroEset)
    attrs <- getDefaultAttrs()
    attrs$graph$rankdir <- "LR"
    plotExpressionGraph(IMCAGraph, IMCAEntrezLink,
                        exprs(fibroEset)[,1],
                        hgu95av2ENTREZID, attrs = attrs)
}
```

---

plotMA-methods *Generate an MA plot*

Description

Generate a plot of log fold change versus mean expression (MA plot)

Usage

```
## S4 method for signature 'data.frame'
plotMA( object, ylim = NULL,
  colNonSig = "gray32", colSig = "red3", colLine = "#ff000080",
  log = "x", cex=0.45, xlab="mean expression", ylab="log fold change", ... )
```

Arguments

object	A data.frame with (at least) three columns, the first containing the mean expression values (for the x-axis), the second the logarithmic fold change (for the-y axis) and the third a logical vector indicating significance (for the colouring of the dots).
ylim	The limits for the y-axis. If missing, an attempt is made to choose a sensible value. Dots exceeding the limits will be displayed as triangles at the limits, pointing outwards.
colNonSig	colour to use for non-significant data points.
colSig	colour to use for significant data points.
colLine	colour to use for the horizontal (y=0) line.
log	which axis/axes should be logarithmic; will be passed to plot.
cex	The cex parameter for plot.
xlab	The x-axis label.
ylab	The y-axis label.
...	Further parameters to be passed through to plot.

Examples

```
plotMA(
  data.frame(
    `M` = exp(rexp(1000)),
    `A` = rnorm(1000) -> tmp,
    `isde` = abs(tmp)>2)
)
```

---

savepng *Save the contents of the current graphics device to a file*

Description

Save the contents of the current graphics device to file

Usage

```
savepdf(fn, dir, width=6, asp=1)
saveeps(fn, dir, width=6, asp=1)
savepng(fn, dir, width=480, asp=1)
savetiff(fn, dir, density=360, keeppdf=TRUE, ...)
```

Arguments

fn	character: name of the output file (without extension). An extension .pdf, .eps, .png, or .tiff will be added automatically.
dir	character: directory to which the file should be written.
width	numeric: width of the image in pixels (png) or inches (pdf, eps).
asp	numeric: aspect ratio; height=width*asp.
density	pixels per inch (see Details).
keeppdf	Should the intermediate PDF file (see Details) be kept? If FALSE, it is deleted before the function returns.
...	Further arguments that are passed on to savepdf (see Details).

Details

The functions are called for their side effect, writing a graphics file.

savepdf, savepng, and saveeps use the devices pdf, png, and postscript, respectively.

There is currently no TIFF device for R, so savetiff works differently. It relies on the external tool convert from the ImageMagick software package. First, savetiff produces a PDF files with savepdf, then uses system to invoke convert with the parameter density. savetiff does **not** check for the existence of convert or the success of the system call, and returns silently no matter what.

Value

Character: name of the file that was written.

Author(s)

Wolfgang Huber http://www.dkfz.de/abt0840/whuber

See Also

dev.copy, pdf, png, postscript

Examples

```
x = seq(0, 20*pi, len=1000)
plot(x*sin(x), x*cos(x), type="l")

try({   ## on some machines, some of the devices may not be available
  c(
    savepdf("spiral", dir=tempdir()),
    savepng("spiral", dir=tempdir()),
    saveeps("spiral", dir=tempdir()),
    savetiff("spiral", dir=tempdir())
   )
})
```

Index

∗ **IO**
openHtmlPage, 18
∗ **datasets**
expressionSet133a, 8
∗ **dplot**
Makesense, 15
∗ **error**
savepng, 23
∗ **graphs**
plotExpressionGraph, 21
∗ **hplot**
amplicon.plot, 4
histStack, 11
multiecdf, 16
plotChr, 19
plotExpressionGraph, 21
∗ **manip**
GetColor, 9
imageMap-methods, 12
∗ **programming**
savepng, 23
∗ **utilities**
alongChrom, 2
cColor, 5
cPlot, 6
cScale, 7
make.chromOrd, 14
plotExpressionGraph, 21

alongChrom, 2
amplicon.plot, 4, *14*

barplot, *11*
boxplot, *16*, *18*
buildACMainLabel *(*alongChrom*)*, 2

cColor, 5, *7*
closeHtmlPage, *13*
closeHtmlPage *(*openHtmlPage*)*, 18
connections, *19*
cPlot, 6, *6*, *8*
cScale, 7, *7*
cullACXPoints *(*alongChrom*)*, 2

dChip.colors *(*GetColor*)*, 9
defMapFun *(*plotExpressionGraph*)*, 21
density, *17*, *18*
dev.copy, *24*
dispACXaxis *(*alongChrom*)*, 2
doACCumPlot *(*alongChrom*)*, 2
doACImagePlot *(*alongChrom*)*, 2
doACLocalPlot *(*alongChrom*)*, 2
doACMatPlot *(*alongChrom*)*, 2

ecdf, *18*
emptyACPlot *(*alongChrom*)*, 2
esApply, *4*, *5*
expressionSet133a, 8

fixACPhysPoints *(*alongChrom*)*, 2

getACClosestPos *(*alongChrom*)*, 2
getACDataEnv *(*alongChrom*)*, 2
getACExprs *(*alongChrom*)*, 2
getACGeneSyms *(*alongChrom*)*, 2
getACPlotLabs *(*alongChrom*)*, 2
getACStrandVals *(*alongChrom*)*, 2
GetColor, 9
getPlotExpressionColors
*(*plotExpressionGraph*)*, 21
greenred.colors *(*GetColor*)*, 9
grid.rect, *10*, *11*
grid.text, *11*
groupedHeatmap, 10

highlightACDups *(*alongChrom*)*, 2
hist, *11*
histStack, 11

imageMap, *13*
imageMap *(*imageMap-methods*)*, 12

imageMap,matrix,connection,list,character-method
(imageMap-methods), 12
imageMap,matrix-method
(imageMap-methods), 12
imageMap-methods, 12
IMCAEntrezLink (plotExpressionGraph), 21

legend, *17*
limitACXRange (alongChrom), 2
log2, *21*

make.chromOrd, *5*, 14
Makesense, 15, *20*
Makesense,ExpressionSet,character-method
(Makesense), 15
Makesense,ExpressionSet,missing-method
(Makesense), 15
Makesense,matrix,character-method
(Makesense), 15
multidensity (multiecdf), 16
multiecdf, 16

openHtmlPage, *13*, 18

pdf, *24*
plot, *23*
plot.graph, *22*
plotChr, *16*, 19
plotExpressionGraph, 21
plotMA (plotMA-methods), 22
plotMA,data.frame-method
(plotMA-methods), 22
plotMA,ExonCountSet-method
(plotMA-methods), 22
plotMA-methods, 22
plotPlate, *13*
png, *13*, *24*
postscript, *24*

saveeps (savepng), 23
savepdf (savepng), 23
savepng, 23
savetiff (savepng), 23
scaleACData (alongChrom), 2
system, *24*

topo.colors, *9*

writeLines, *13*